Foundation of Cryptography, Lecture 11
Black-Box Impossibility Results

Iftach Haitner, Tel Aviv University

Tel Aviv University.

June 10, 2014

Motivating example: Basing Key-Agreement on OWFs

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.
- Can we base KA on OWFs/OWPs?

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.
- Can we base KA on OWFs/OWPs?
- Proving unconditional negative result seems beyond reach.

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.
- Can we base KA on OWFs/OWPs?
- Proving unconditional negative result seems beyond reach.

  Assume RSA assumption holds.

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.
- Can we base KA on OWFs/OWPs?
- Proving unconditional negative result seems beyond reach.

  Assume RSA assumption holds.

  $\Longrightarrow$ key-agreement protocols exist.

Motivating example: Basing Key-Agreement on OWFs

- Key-Agreement protocols (KA) can be based on the existence of TDP, RSA or discrete log assumptions, and ...
- We don't know how to base KA on the existence of OWFs/OWPs.
- Can we base KA on OWFs/OWPs?
- Proving unconditional negative result seems beyond reach.

  Assume RSA assumption holds.

  $\Longrightarrow$ key-agreement protocols exist.

  $\Longrightarrow$ OWFs imply the existence of key-agreement protocols in a trivial sense.

(Fully) Black-box constructions

(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of B from A)

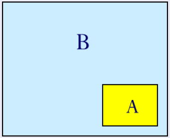
Black-box **construction**:

(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of B from A)

Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements B for any algorithm O implementing A

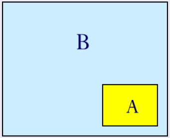


(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of B from A)

Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements B for any algorithm O implementing A



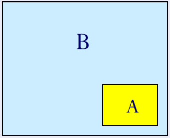
Black-box **proof of security**:

(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of *B* from *A*)

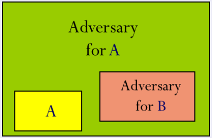
Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements *B* for any algorithm O implementing *A*



Black-box **proof of security**:

A oracle-aided PPT R such that $\mathrm{R}^{\mathrm{O},\mathrm{D}}$ breaks *B*, for any algorithms O implementing *A*, and D breaking *B*.

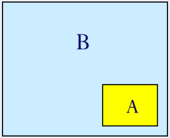


(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of *B* from *A*)

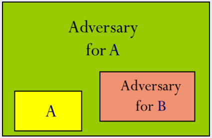
Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements *B* for any algorithm O implementing *A*



Black-box **proof of security**:

A oracle-aided PPT R such that $\mathsf{R}^{\mathsf{O},\mathsf{D}}$ breaks *B*, for any algorithms O implementing *A*, and D breaking *B*.



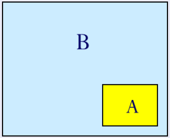
- Fully-black-box constructions relativize: hold relative to any oracle.

(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of B from A)

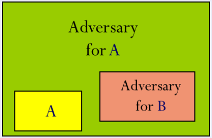
Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements B for any algorithm O implementing A



Black-box **proof of security**:

A oracle-aided PPT R such that $\mathrm{R}^{\mathrm{O},\mathrm{D}}$ breaks B, for any algorithms O implementing A, and D breaking B.



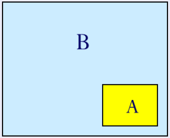
- Fully-black-box constructions relativize: hold relative to any oracle.
- Most constructions in cryptography are (fully) black-box, e.g., pseudorandom generator from OWF.

(Fully) Black-box constructions

Definition 1 (A fully Black-box construction of B from A)

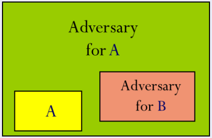
Black-box **construction**:

A oracle-aided PPT I such that I^{O} implements B for any algorithm O implementing A



Black-box **proof of security**:

A oracle-aided PPT R such that $\mathsf{R}^{\mathsf{O},\mathsf{D}}$ breaks B, for any algorithms O implementing A, and D breaking B.



- Fully-black-box constructions relativize: hold relative to any oracle.
- Most constructions in cryptography are (fully) black-box, e.g., pseudorandom generator from OWF.
- Few "non black-box" techniques that apply in restricted settings (typically using ZK proofs)

Approach for proving BB impossibility result

Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.

Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.

Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.

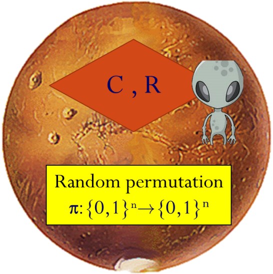
Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.

I^{π} is an efficient KA protocol and $\mathsf{R}^{\pi,\mathsf{D}}$ should invert π for any D breaking I^{π}.

Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.

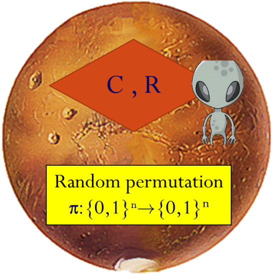


I^{π} is an efficient KA protocol and $\mathsf{R}^{\pi,\mathsf{D}}$ should invert π for any D breaking I^{π}.

Assume $\exists$ (even inefficient) algorithm D that breaks any efficient KA with oracle access to π, but is not useful for inverting π.

Approach for proving BB impossibility result

Assume $\exists$ fully-BB construction (I, R) of a KA from OWP.



I^{π} is an efficient KA protocol and $\mathsf{R}^{\pi,\mathsf{D}}$ should invert π for any D breaking I^{π}.

Assume $\exists$ (even inefficient) algorithm D that breaks any efficient KA with oracle access to π, but is not useful for inverting π.

This yields a contradiction, implying that (I, R) does not exist.

Section 1

Random Permutations

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log \binom{N}{a}$ bits to describe $\mathcal{S}$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log\binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)! \cdot a!} \leq (\frac{eb}{a})^a$

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log\binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)! \cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log\binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)!\cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log \binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)!\cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log\binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)!\cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.
  - The latter is larger in for $\alpha > \frac{1}{2}$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log\binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)!\cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.
  - The latter is larger in for $\alpha > \frac{1}{2}$.
- We are in the number of M-size oracle-circuits .

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log \binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)! \cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.
  - The latter is larger in for $\alpha > \frac{1}{2}$.
- We are in the number of M-size oracle-circuits .

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a? There are $\binom{N}{a}$ such sets, so it takes $\log \binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)! \cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.
  - The latter is larger in for $\alpha > \frac{1}{2}$.
- We are in the number of M-size oracle-circuits .

Claim 2

The number of M-size oracle-circuits mapping n-bit strings to n-bit strings, with oracle access to a function n-bit strings to n-bit strings, is at most $2^{2M+(M+1)n(\log(Mn+n)+1)}$.

Before we begin

- Let Π_n — set of all permutations over $\{0,1\}^n$, and let $N = 2^n$.
- $|\Pi_n| = N! \implies$ it takes $\log(N!)$ bits to describe $\pi \in \Pi_n$.
- How many bits it takes to describe $\mathcal{S} \subseteq \{0,1\}^n$ of (known) size a?
  There are $\binom{N}{a}$ such sets, so it takes $\log \binom{N}{a}$ bits to describe $\mathcal{S}$.
- For integer $b \geq a$: $a! \geq (\frac{a}{e})^a$ and $\binom{b}{a} := \frac{b!}{(b-a)!\cdot a!} \leq (\frac{eb}{a})^a$
- For $a = 2^{\alpha n}$-size set $\mathcal{S} \subseteq \{0,1\}^n$:
  - It takes at most $a \cdot ((1-\alpha)n + O(1))$ bits to describe .
  - It takes at least $a(\alpha n - O(1))$ bits to describe a permutation over $\mathcal{S}$.
  - The latter is larger in for $\alpha > \frac{1}{2}$.
- We are in the number of M-size oracle-circuits .

Claim 2

The number of M-size oracle-circuits mapping n-bit strings to n-bit strings, with oracle access to a function n-bit strings to n-bit strings, is at most $2^{2M+(M+1)n(\log(Mn+n)+1)}$.

Proof: ?

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ and $2^{n/5}$-query circuit D,

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ *and* $2^{n/5}$*-query circuit* D*,*

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ and $2^{n/5}$-query circuit D,

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.
- Constants are somewhat arbitrary and non tight.

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ and $2^{n/5}$-query circuit D,

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.
- Constants are somewhat arbitrary and non tight.
- By Claim 2 the number of $2^{n/5}$-size circuits (of the right form) is bounded by $2^{\tilde{O}(2^{n/5})}$.

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ *and* $2^{n/5}$*-query circuit* D*,*

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.
- Constants are somewhat arbitrary and non tight.
- By Claim 2 the number of $2^{n/5}$-size circuits (of the right form) is bounded by $2^{\tilde{O}(2^{n/5})}$.

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ *and* $2^{n/5}$*-query circuit* D*,*

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.
- Constants are somewhat arbitrary and non tight.
- By Claim 2 the number of $2^{n/5}$-size circuits (of the right form) is bounded by $2^{\tilde{O}(2^{n/5})}$. Thus Thm 3 yields that

$$\Pr_{\pi \leftarrow \Pi_n}\left[\exists\ 2^{n/5}\text{-size circuit } \mathsf{C} \text{ with } \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{n/2}}$$

Random permutations are hard to invert

Theorem 3 (Gennarro-Tevisan, '01)

For any large enough $n \in \mathbb{N}$ and $2^{n/5}$-query circuit D,

$$\Pr_{\pi \leftarrow \Pi_n}\left[\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{\frac{3}{5}n}/2}$$

- In words: Random permutations are (extremely) hard even for exponential-size circuits.
- Constants are somewhat arbitrary and non tight.
- By Claim 2 the number of $2^{n/5}$-size circuits (of the right form) is bounded by $2^{\tilde{O}(2^{n/5})}$. Thus Thm 3 yields that
$$\Pr_{\pi \leftarrow \Pi_n}\left[\exists\ 2^{n/5}\text{-size circuit } \mathsf{C} \text{ with } \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}\right] \leq 2^{-2^{n/2}}$$
- In words: Random permutations are (extremely) hard simultaneously, for all exponential-size circuits.

Proving GT theorem (Thm 3)

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^{\pi}(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^{\pi}(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

- The description of π using $\mathsf{Enc}(\pi)$ “saves" $\log(a!) - \log \binom{N}{a}$ bits.

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^{\pi}(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

- The description of π using $\mathsf{Enc}(\pi)$ “saves" $\log(a!) - \log \binom{N}{a}$ bits.
- Let D be a $2^{n/5}$-query circuit.

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^{\pi}(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

- The description of π using $\mathsf{Enc}(\pi)$ “saves" $\log(a!) - \log \binom{N}{a}$ bits.
- Let D be a $2^{n/5}$-query circuit.

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^{\pi}(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

- The description of π using $\mathsf{Enc}(\pi)$ “saves" $\log(a!) - \log \binom{N}{a}$ bits.
- Let D be a $2^{n/5}$-query circuit. Lemma 4 yields that the fraction of $\pi \in \Pi_n$ with $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}$, is (for large enough n) at most

$$\frac{(N - 2^{\frac{3}{5}n})! \cdot \binom{N}{2^{\frac{3}{5}n}}^2}{N!} = \frac{\binom{N}{2^{\frac{3}{5}n}}}{2^{\frac{3}{5}n}!} \leq 2^{-2^{\frac{3}{5}n}/2},$$

Proving GT theorem (Thm 3)

Lemma 4 (compression lemma)

For any q-query circuit D and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}^\pi(\pi(x)) = x] > \varepsilon$, then

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \leq \log((2^n - a)!) + 2 \cdot \log \binom{N}{a}$, *for some* $a \geq \frac{\varepsilon N}{q+1}$.

- The description of π using $\mathsf{Enc}(\pi)$ “saves" $\log(a!) - \log \binom{N}{a}$ bits.
- Let D be a $2^{n/5}$-query circuit. Lemma 4 yields that the fraction of $\pi \in \Pi_n$ with $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > 2^{-n/5}$, is (for large enough n) at most

$$\frac{(N - 2^{\frac{3}{5}n})! \cdot \binom{N}{2^{\frac{3}{5}n}}^2}{N!} = \frac{\binom{N}{2^{\frac{3}{5}n}}}{2^{\frac{3}{5}n}!} \leq 2^{-2^{\frac{3}{5}n}/2},$$

proving Thm 3.

Proving Compression Lemma (Lemma 4)

Let D be q-query circuit, and let π be such that $\Pr_{x \leftarrow \{0,1\}^n}[D(\pi(x)) = x] > \varepsilon$.

Proving Compression Lemma (Lemma 4)

Let D be q-query circuit, and let π be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > \varepsilon$.

Construction 5 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon \mathsf{D}^{\pi}(\pi(x)) = \pi\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lexicographic element in $\mathcal{I}$.
   - **(a)** Add y to $\mathcal{Y}$.
   - **(b)** Remove y and all π-queries $\mathsf{D}^{\pi}(y)$ makes, from $\mathcal{I}$.

Proving Compression Lemma (Lemma 4)

Let D be q-query circuit, and let π be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > \varepsilon$.

Construction 5 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n : \mathsf{D}^\pi(\pi(x)) = \pi\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lexicographic element in $\mathcal{I}$.
   - **(a)** Add y to $\mathcal{Y}$.
   - **(b)** Remove y and all π-queries $\mathsf{D}^\pi(y)$ makes, from $\mathcal{I}$.

Algorithm 6 ($\mathsf{Enc}(\pi)$)

Output (description of) $\mathcal{Y}$ and $\mathcal{V} = \{(x, \pi(x)) : \pi(x) \notin \mathcal{Y}\}$.

Proving Compression Lemma (Lemma 4)

Let D be q-query circuit, and let π be such that $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{D}(\pi(x)) = x] > \varepsilon$.

Construction 5 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon \mathsf{D}^{\pi}(\pi(x)) = \pi\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lexicographic element in $\mathcal{I}$.
   - **(a)** Add y to $\mathcal{Y}$.
   - **(b)** Remove y and all π-queries $\mathsf{D}^{\pi}(y)$ makes, from $\mathcal{I}$.

Algorithm 6 ($\mathsf{Enc}(\pi)$)

Output (description of) $\mathcal{Y}$ and $\mathcal{V} = \{(x, \pi(x)) \colon \pi(x) \notin \mathcal{Y}\}$.

(Under proper encoding) $|\mathsf{Enc}(\pi)| \leq \log((N-a)!) + 2 \cdot \log\binom{N}{a}$ for $a = |Y| \geq \frac{\varepsilon N}{q+1}$.

Proving Compression Lemma (Lemma 4)

Let D be q-query circuit, and let π be such that $\Pr_{x \leftarrow \{0,1\}^n}[D(\pi(x)) = x] > \varepsilon$.

Construction 5 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon D^{\pi}(\pi(x)) = \pi\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lexicographic element in $\mathcal{I}$.
   (a) Add y to $\mathcal{Y}$.
   (b) Remove y and all π-queries $D^{\pi}(y)$ makes, from $\mathcal{I}$.

Algorithm 6 (Enc(π))

Output (description of) $\mathcal{Y}$ and $\mathcal{V} = \{(x, \pi(x)) \colon \pi(x) \notin \mathcal{Y}\}$.

(Under proper encoding) $|\text{Enc}(\pi)| \leq \log((N-a)!) + 2 \cdot \log \binom{N}{a}$ for $a = |Y| \geq \frac{\varepsilon N}{q+1}$.

Algorithm 7 (Dec($\mathcal{Y}, \mathcal{V}$))

For all $y \in \mathcal{Y}$ in lex. order:

1. Emulate $D^{\pi}(y)$.
2. If D makes a π-query x that is undefined in $\mathcal{V}$, add (x, y) to $\mathcal{V}$.
   Otherwise, add $(D^{\pi, \text{Sam}_r^{\pi}}(y), y)$ to $\mathcal{V}$.

Use $\mathcal{V}$ to reconstruct π.

Remarks

- The short description argument is an incredibly useful paradigm (see next).

Remarks

- The short description argument is an incredibly useful paradigm (see next).
- Immediately yields same result for algorithms (replacing size with running time).

Remarks

- The short description argument is an incredibly useful paradigm (see next).
- Immediately yields same result for algorithms (replacing size with running time).
- Alternative compression argument.

Remarks

- The short description argument is an incredibly useful paradigm (see next).
- Immediately yields same result for algorithms (replacing size with running time).
- Alternative compression argument.
- Similar results can be proven for random variants of OWF, TDP, CRH.

Section 2

BB Impossibility for Efficient OWF based PRG

BB Impossibility for OWF based PRG

Definition 8 (pseudorandom generators (PRGs))

Poly-time $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is a pseudorandom generator, if

- G is length extending (i.e., $\ell(n) > n$ for any n)
- $G(U_n)$ is pseudorandom (i.e., $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$)

BB Impossibility for OWF based PRG

Definition 8 (pseudorandom generators (PRGs))

Poly-time $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is a pseudorandom generator, if

- G is length extending (i.e., $\ell(n) > n$ for any n)
- $G(U_n)$ is pseudorandom (i.e., $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$)

We focus on BB constructions of efficient length-doubling PRGs.

BB Impossibility for OWF based PRG

Definition 8 (pseudorandom generators (PRGs))

Poly-time $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is a pseudorandom generator, if

- G is length extending (i.e., $\ell(n) > n$ for any n)
- $G(U_n)$ is pseudorandom (i.e., $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$)

We focus on BB constructions of efficient length-doubling PRGs.

Theorem 9

In any fully-BB construction of length-doubling PRG over n-bits string from OWP over $\{0,1\}^n$, the construction makes $\Omega(n/\log n)$ oracle calls.

BB Impossibility for OWF based PRG

Definition 8 (pseudorandom generators (PRGs))

Poly-time $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is a pseudorandom generator, if

- G is length extending (i.e., $\ell(n) > n$ for any n)
- $G(U_n)$ is pseudorandom (i.e., $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$)

We focus on BB constructions of efficient length-doubling PRGs.

Theorem 9

In any fully-BB construction of length-doubling PRG over n-bits string from OWP over $\{0,1\}^n$, the construction makes $\Omega(n/\log n)$ oracle calls.

- Matches known upper bounds.

BB Impossibility for OWF based PRG

Definition 8 (pseudorandom generators (PRGs))

Poly-time $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is a pseudorandom generator, if

- G is length extending (i.e., $\ell(n) > n$ for any n)
- $G(U_n)$ is pseudorandom (i.e., $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$)

We focus on BB constructions of efficient length-doubling PRGs.

Theorem 9

In any fully-BB construction of length-doubling PRG over n-bits string from OWP over $\{0,1\}^n$, the construction makes $\Omega(n/\log n)$ oracle calls.

- Matches known upper bounds.
- Without the restriction on the OWP input length, yields an optimal $n^{\Omega(1)}/\log n$ bound.

Proving Thm 9

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.

  We assume wlg. that I makes distinct queries.(?)

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.
  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G \colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.

  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G \colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.
  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G \colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

1. Emulate $\mathrm{C}^{\pi}(x_{1,\ldots,n})$, while answering the i'th query z of I to π, with $x_{n+i\cdot t+1,\ldots,n+(i+1)\cdot t} \circ z_{t+1,\ldots,n}$.

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.
  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G \colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

1. Emulate $\mathrm{C}^{\pi}(x_{1,\ldots,n})$, while answering the i'th query z of I to π, with $x_{n+i\cdot t+1,\ldots,n+(i+1)\cdot t} \circ z_{t+1,\ldots,n}$.
2. Output the same output that I does.

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.

  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G\colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

1. Emulate $\mathrm{C}^\pi(x_{1,\ldots,n})$, while answering the i'th query z of I to π, with $x_{n+i\cdot t+1,\ldots,n+(i+1)\cdot t} \circ z_{t+1,\ldots,n}$.
2. Output the same output that I does.

- Let $\Pi_{n,t}$ be the set of all permutations over $\{0,1\}^n$ that are identity over the last $n-t$ bits (i.e., $\pi(x)_{n-t+1,\ldots,n} = x_{n-t+1,\ldots,n}$).

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.
  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G \colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

1. Emulate $\mathrm{C}^{\pi}(x_{1,\ldots,n})$, while answering the i'th query z of I to π, with $x_{n+i\cdot t+1,\ldots,n+(i+1)\cdot t} \circ z_{t+1,\ldots,n}$.
2. Output the same output that I does.

- Let $\Pi_{n,t}$ be the set of all permutations over $\{0,1\}^n$ that are identity over the last $n-t$ bits (i.e., $\pi(x)_{n-t+1,\ldots,n} = x_{n-t+1,\ldots,n}$).

Proving Thm 9

- Let (I, R) be a fully-BB reduction of a $q(n) \in o(n/\log n)$-query, length-doubling PRG over $\{0,1\}^n$, to OWP over $\{0,1\}^n$.
  We assume wlg. that I makes distinct queries.(?)
- For $t = t(n) = \lceil n/2q(n) \rceil$, consider the following generator $G\colon \{0,1\}^{\frac{3}{2}n} \mapsto \{0,1\}^{2n}$:

Algorithm 10 ($G(x)$)

1. Emulate $\mathrm{C}^{\pi}(x_{1,\ldots,n})$, while answering the i'th query z of I to π, with $x_{n+i\cdot t+1,\ldots,n+(i+1)\cdot t} \circ z_{t+1,\ldots,n}$.
2. Output the same output that I does.

- Let $\Pi_{n,t}$ be the set of all permutations over $\{0,1\}^n$ that are identity over the last $n-t$ bits (i.e., $\pi(x)_{n-t+1,\ldots,n} = x_{n-t+1,\ldots,n}$).

Claim 11

$G(U_{3n/2}) \equiv (\mathrm{I}^{\pi}(U_n))_{\pi \leftarrow \Pi_{n,t}}$.

Proving Thm 9, cont.

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\Longrightarrow$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathrm{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathrm{D}(U_{2n}) = 1] > \frac{1}{2}$

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\implies$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\implies$ $\Pr_{\pi \leftarrow \Pi_{t,n}}\left[\Pr[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1-2^{-n/4} > \frac{1}{2}$. (?)

  $\implies$ wlg. $\Pr_{\pi\leftarrow\Pi_{t,n}}[\mathrm{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathrm{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\implies \Pr_{\pi\leftarrow\Pi_{t,n}}\left[\Pr[\mathrm{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathrm{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

  $\implies \Pr_{\pi\leftarrow\Pi_{t,n}}\left[\mathrm{R}^{\pi,\mathrm{D}} \text{ inverts } \pi \text{ with non-negligible prob.}\right] \geq \frac{1}{4}$

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\Longrightarrow$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\Longrightarrow$ $\Pr_{\pi \leftarrow \Pi_{t,n}}\left[\Pr[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

  $\Longrightarrow$ $\Pr_{\pi \leftarrow \Pi_{t,n}}\left[\mathsf{R}^{\pi,\mathsf{D}} \text{ inverts } \pi \text{ with non-negligible prob.}\right] \geq \frac{1}{4}$

- Let $n' = t(n) \in \omega(\log n)$.

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\implies$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\implies \Pr_{\pi \leftarrow \Pi_{t,n}}\left[\Pr[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

  $\implies \Pr_{\pi \leftarrow \Pi_{t,n}}\left[\mathsf{R}^{\pi,\mathsf{D}}\right.$ inverts π with non-negligible prob.$\left.\right] \geq \frac{1}{4}$

- Let $n' = t(n) \in \omega(\log n)$.
- By the above, exists $2^{o(n')}$-query circuit R', such that

$$\Pr_{\pi \leftarrow \Pi_{n'}}[\mathsf{R}'^{\pi} \text{ inverts } \pi \text{ with non-negligible prob.}] \geq \frac{1}{4},$$

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\implies$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathrm{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathrm{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\implies$ $\Pr_{\pi \leftarrow \Pi_{t,n}}\left[\Pr[\mathrm{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathrm{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

  $\implies$ $\Pr_{\pi \leftarrow \Pi_{t,n}}\left[\mathrm{R}^{\pi,\mathrm{D}} \text{ inverts } \pi \text{ with non-negligible prob.}\right] \geq \frac{1}{4}$
- Let $n' = t(n) \in \omega(\log n)$.
- By the above, exists $2^{o(n')}$-query circuit R', such that

$$\Pr_{\pi \leftarrow \Pi_{n'}}[\mathrm{R}'^{\pi} \text{ inverts } \pi \text{ with non-negligible prob.}] \geq \frac{1}{4},$$

Proving Thm 9, cont.

- $\exists$ algorithm D that distinguishes $G(U_{3n/2})$ from U_{2n} with advantage $1 - 2^{-n/4} > \frac{1}{2}$. (?)

  $\implies$ wlg. $\Pr_{\pi \leftarrow \Pi_{t,n}}[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{2}$

  $\implies \Pr_{\pi \leftarrow \Pi_{t,n}}\left[\Pr[\mathsf{D}(\mathrm{I}^{\pi}(U_n)) = 1] - \Pr[\mathsf{D}(U_{2n}) = 1] > \frac{1}{4}\right] \geq \frac{1}{4}$

  $\implies \Pr_{\pi \leftarrow \Pi_{t,n}}\left[\mathsf{R}^{\pi,\mathsf{D}} \text{ inverts } \pi \text{ with non-negligible prob.}\right] \geq \frac{1}{4}$
- Let $n' = t(n) \in \omega(\log n)$.
- By the above, exists $2^{o(n')}$-query circuit R', such that

$$\Pr_{\pi \leftarrow \Pi_{n'}}[\mathsf{R}'^{\pi} \text{ inverts } \pi \text{ with non-negligible prob.}] \geq \frac{1}{4},$$

  in contradiction to Thm 3.

Remarks

- We showed a lower bound on the efficiency of fully-BB constructions of length-doubling PRG from OWPs.

Remarks

- We showed a lower bound on the efficiency of fully-BB constructions of length-doubling PRG from OWPs.
- Actually we ruled out a **less restricted** type of BB-construction, called weakly-BB construction:

  If O is a secure implementation of A, then I^{O} is a secure implementation of B against adversaries with no access direct to O.

Remarks

- We showed a lower bound on the efficiency of fully-BB constructions of length-doubling PRG from OWPs.
- Actually we ruled out a **less restricted** type of BB-construction, called weakly-BB construction:

  If O is a secure implementation of A, then I^{O} is a secure implementation of B against adversaries with no access direct to O.
- Results can be easily extended to OWFs/TDPs.

Remarks

- We showed a lower bound on the efficiency of fully-BB constructions of length-doubling PRG from OWPs.
- Actually we ruled out a **less restricted** type of BB-construction, called weakly-BB construction:

  If O is a secure implementation of A, then I^{O} is a secure implementation of B against adversaries with no access direct to O.
- Results can be easily extended to OWFs/TDPs.
- Using similar means, one can prove lower bound on fully-BB constructions of encryption schemes, signature schemes and universal-one-way-hash-functions (UOWHFs), from OWFs/OWPs/TDPs

Section 3

BB Impossibility for Basing CRH on OWF

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} \left[x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')\right] = \operatorname{neg}(n)$$

for any PPT A.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \land h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Theorem 13

There exists no fully BB-construction of CRH from OWP.

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Theorem 13

There exists no fully BB-construction of CRH from OWP.

- Seems harder to prove:

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Theorem 13

There exists no fully BB-construction of CRH from OWP.

- Seems harder to prove:

Basing CRH on OWF

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (x,x') \leftarrow \mathsf{A}(1^n,h)}} [x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- In a BB construction of CRH family, both the sampling and evaluation algorithms makes use of the oracle.
- wlg. the sampling algorithm outputs a string h, independent of the oracle, and $h \in \mathcal{H}_n$ is an oracle circuit.
- For simplicity, assume that $h \in \mathcal{H}_n$ only queries the oracle on inputs of length n.

Theorem 13

There exists no fully BB-construction of CRH from OWP.

- Seems harder to prove: CRH exists relative to random permutations!

Proving Thm 13

Proving Thm 13

Fix $n \in \mathbb{N}$.

Proving Thm 13

Fix $n \in \mathbb{N}$.

We construct an (inefficient) algorithm Sam, that finds collision in any CRH, but non-useful for inverting random permutation.

Proving Thm 13

Fix $n \in \mathbb{N}$.

We construct an (inefficient) algorithm Sam, that finds collision in any CRH, but non-useful for inverting random permutation.

Algorithm 14 (Sam^π)

Input: An n-bit input circuit C.
Oracle: $\pi \in \Pi_n$.

1. Sample $x \leftarrow \{0,1\}^n$.
2. Find the first (in a random order) random $x' \in \{0,1\}^n$ with $\mathsf{C}^\pi(x) = \mathsf{C}^\pi(x')$.
3. Return (x, x').

Proving Thm 13

Fix $n \in \mathbb{N}$.

We construct an (inefficient) algorithm Sam, that finds collision in any CRH, but non-useful for inverting random permutation.

Algorithm 14 (Sam^{π})

Input: An n-bit input circuit C.
Oracle: $\pi \in \Pi_n$.

1. Sample $x \leftarrow \{0,1\}^n$.
2. Find the first (in a random order) random $x' \in \{0,1\}^n$ with $\mathsf{C}^{\pi}(x) = \mathsf{C}^{\pi}(x')$.
3. Return (x, x').

- Let Sam_r be the variant of Sam whose coins are fixed to r.

Proving Thm 13

Fix $n \in \mathbb{N}$.

We construct an (inefficient) algorithm Sam, that finds collision in any CRH, but non-useful for inverting random permutation.

Algorithm 14 (Sam^{π})

Input: An n-bit input circuit C.
Oracle: $\pi \in \Pi_n$.

1. Sample $x \leftarrow \{0,1\}^n$.
2. Find the first (in a random order) random $x' \in \{0,1\}^n$ with $\mathsf{C}^{\pi}(x) = \mathsf{C}^{\pi}(x')$.
3. Return (x, x').

- Let Sam_r be the variant of Sam whose coins are fixed to r.
- In the actual implementation Sam uses independent randomness per input query C.

No CRH relative to Sam

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^{\pi}(h)}[x \neq x' \wedge h^{\pi}(x) = h^{\pi}(x')] \geq \frac{1}{4}$.

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^{\pi}(h)}\left[x \neq x' \wedge h^{\pi}(x) = h^{\pi}(x')\right] \geq \frac{1}{4}$.

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^{\pi}(h)}[x \neq x' \wedge h^{\pi}(x) = h^{\pi}(x')] \geq \frac{1}{4}$.

Proof:

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^{\pi}(h)}\left[x \neq x' \wedge h^{\pi}(x) = h^{\pi}(x')\right] \geq \frac{1}{4}$.

Proof: It suffices to prove that for any length decreasing function g over $\{0,1\}^n$, $\Pr_{x \leftarrow \{0,1\}^n}\left[\left|g^{-1}(g(x))\right| = 1\right] \leq \frac{1}{2}$.$\square$

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^{\pi}(h)}[x \neq x' \wedge h^{\pi}(x) = h^{\pi}(x')] \geq \frac{1}{4}$.

Proof: It suffices to prove that for any length decreasing function g over $\{0,1\}^n$, $\Pr_{x \leftarrow \{0,1\}^n}\left[\left|g^{-1}(g(x))\right| = 1\right] \leq \frac{1}{2}$.$\square$

The following algorithm breaks the collision resistance of any Black-box construction of a CRH from OWP.

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^\pi(h)}[x \neq x' \wedge h^\pi(x) = h^\pi(x')] \geq \frac{1}{4}$.

Proof: It suffices to prove that for any length decreasing function g over $\{0,1\}^n$, $\Pr_{x \leftarrow \{0,1\}^n}\left[\left|g^{-1}(g(x))\right| = 1\right] \leq \frac{1}{2}$.$\square$

The following algorithm breaks the collision resistance of any Black-box construction of a CRH from OWP.

Algorithm 16 ($\mathsf{D}^{\mathsf{Sam}^\pi}$)

On input $h \in \mathcal{H}_n$, return $\mathsf{Sam}^\pi(h)$.

No CRH relative to Sam

Let $\mathcal{H}_n$ be a length-decreasing oracle-aided circuit family.

Claim 15

For any $h \in \mathcal{H}_n$ and $\pi \in \Pi_n$: $\Pr_{(x,x') \leftarrow \mathsf{Sam}^\pi(h)}[x \neq x' \wedge h^\pi(x) = h^\pi(x')] \geq \frac{1}{4}$.

Proof: It suffices to prove that for any length decreasing function g over $\{0,1\}^n$, $\Pr_{x \leftarrow \{0,1\}^n}\left[\left|g^{-1}(g(x))\right| = 1\right] \leq \frac{1}{2}$. $\square$

The following algorithm breaks the collision resistance of any Black-box construction of a CRH from OWP.

Algorithm 16 ($\mathsf{D}^{\mathsf{Sam}^\pi}$)

On input $h \in \mathcal{H}_n$, return $\mathsf{Sam}^\pi(h)$.

Random permutations are hard relative to Sam

Random permutations are hard relative to Sam

Proof via the GT paradigm,

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}$, via Sam_r^π, might make more than 2^n queries...

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}$, via Sam_r^π, might make more than 2^n queries...

Idea: focus only on the “collision" queries made by Sam_r^π.

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}$, via Sam_r^π, might make more than 2^n queries...

Idea: focus only on the "collision" queries made by Sam_r^π.

Definition 17

The augmented number of queries an oracle-aided circuit/algorithm with Sam-gate does, is the number of queries it makes **directly**, plus twice the number of queries the circuits it **passes** to Sam do.

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $D^{\pi,\mathrm{Sam}_r^\pi}$, via Sam_r^π, might make more than 2^n queries...

Idea: focus only on the “collision" queries made by Sam_r^π.

Definition 17

The augmented number of queries an oracle-aided circuit/algorithm with Sam-gate does, is the number of queries it makes **directly**, plus twice the number of queries the circuits it **passes** to Sam do.

Theorem 18

For any large enough $n \in \mathbb{N}$ and $2^{n/5}/2$-augmented-query circuit D:

$$\Pr_{\pi\leftarrow\Pi_n; r\leftarrow\{0,1\}^*}\left[\Pr_{x\leftarrow\{0,1\}^n}\left[D^{\pi,\mathrm{Sam}_r^\pi}(\pi(x)) = x\right] > 2\cdot 2^{-n/5}\right] \leq 2\cdot 2^{-2^{\frac{3}{5}n}}$$

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}$, via Sam_r^π, might make more than 2^n queries...

Idea: focus only on the "collision" queries made by Sam_r^π.

Definition 17

The augmented number of queries an oracle-aided circuit/algorithm with Sam-gate does, is the number of queries it makes **directly**, plus twice the number of queries the circuits it **passes** to Sam do.

Theorem 18

For any large enough $n \in \mathbb{N}$ and $2^{n/5}/2$-augmented-query circuit D:

$$\Pr_{\pi\leftarrow\Pi_n; r\leftarrow\{0,1\}^*}\left[\Pr_{x\leftarrow\{0,1\}^n}\left[\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = x\right] > 2\cdot 2^{-n/5}\right] \leq 2\cdot 2^{-2^{\frac{3}{5}n}}$$

- Almost the same result as in the non-Sam case.

Random permutations are hard relative to Sam

Proof via the GT paradigm, but $D^{\pi,\mathsf{Sam}_r^{\pi}}$, via Sam_r^{π}, might make more than 2^n queries...

Idea: focus only on the “collision" queries made by Sam_r^{π}.

Definition 17

The augmented number of queries an oracle-aided circuit/algorithm with Sam-gate does, is the number of queries it makes **directly**, plus twice the number of queries the circuits it **passes** to Sam do.

Theorem 18

For any large enough $n \in \mathbb{N}$ and $2^{n/5}/2$-augmented-query circuit D:

$$\Pr_{\pi \leftarrow \Pi_n; r \leftarrow \{0,1\}^*}\left[\Pr_{x \leftarrow \{0,1\}^n}\left[D^{\pi,\mathsf{Sam}_r^{\pi}}(\pi(x)) = x\right] > 2 \cdot 2^{-n/5}\right] \leq 2 \cdot 2^{-2^{\frac{3}{5}n}}$$

- Almost the same result as in the non-Sam case.
- Hence, random permutations are (extremely) hard for exponential-size circuits with oracle access to Sam.

Proving Thm 18

Proving Thm 18

Fix large enough n. The proof follows by the next two claims.

Proving Thm 18

Fix large enough n. The proof follows by the next two claims.

Definition 19

For circuit D, $\pi \in \Pi_n$, $r \in \{0,1\}^*$, an $y \in \{0,1\}^n$, let $\mathsf{hit}^{\pi}_{\mathsf{D};r}(y)$ be one, if $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$ makes a query $(x, x') = \mathsf{Sam}^{\pi}_r(\mathsf{C})$, and either $\mathsf{C}^{\pi}(x)$ or $\mathsf{C}^{\pi}(x')$ query π on $\pi^{-1}(y)$.

Proving Thm 18

Fix large enough n. The proof follows by the next two claims.

Definition 19

For circuit D, $\pi \in \Pi_n$, $r \in \{0,1\}^*$, an $y \in \{0,1\}^n$, let $\mathsf{hit}^{\pi}_{\mathsf{D};r}(y)$ be one, if $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$ makes a query $(x,x') = \mathsf{Sam}^{\pi}_r(\mathsf{C})$, and either $\mathsf{C}^{\pi}(x)$ or $\mathsf{C}^{\pi}(x')$ query π on $\pi^{-1}(y)$.

The following probabilities are over $\pi \leftarrow \Pi_n$, $r \leftarrow \{0,1\}^*$ and $x \leftarrow \{0,1\}^n$.

Proving Thm 18

Fix large enough n. The proof follows by the next two claims.

Definition 19

For circuit D, $\pi \in \Pi_n$, $r \in \{0,1\}^*$, an $y \in \{0,1\}^n$, let $\mathsf{hit}^{\pi}_{\mathsf{D};r}(y)$ be one, if $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$ makes a query $(x,x') = \mathsf{Sam}^{\pi}_r(\mathsf{C})$, and either $\mathsf{C}^{\pi}(x)$ or $\mathsf{C}^{\pi}(x')$ query π on $\pi^{-1}(y)$.

The following probabilities are over $\pi \leftarrow \Pi_n$, $r \leftarrow \{0,1\}^*$ and $x \leftarrow \{0,1\}^n$.

Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \le$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \land \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ for any $\varepsilon \ge 0$.

Proving Thm 18

Fix large enough n. The proof follows by the next two claims.

Definition 19

For circuit D, $\pi \in \Pi_n$, $r \in \{0,1\}^*$, an $y \in \{0,1\}^n$, let $\mathrm{hit}^{\pi}_{\mathsf{D};r}(y)$ be one, if $\mathsf{D}^{\pi,\mathrm{Sam}^{\pi}_r}(y)$ makes a query $(x,x') = \mathrm{Sam}^{\pi}_r(\mathsf{C})$, and either $\mathsf{C}^{\pi}(x)$ or $\mathsf{C}^{\pi}(x')$ query π on $\pi^{-1}(y)$.

The following probabilities are over $\pi \leftarrow \Pi_n$, $r \leftarrow \{0,1\}^*$ and $x \leftarrow \{0,1\}^n$.

Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathrm{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathrm{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathrm{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ for any $\varepsilon \geq 0$.

Claim 21

For any $2^{n/5}/2$-augQuery circuit D:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{D}^{\pi,\mathrm{Sam}^{\pi}_r}(\pi(x)) = x\right] > 2^{-n/5} \wedge \neg\mathrm{hit}^{\pi}_{\mathsf{D};r}\right] \leq 2^{-2^{3n/5}}$.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:

$$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$$

$$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$$ **for any $\varepsilon \geq 0$.**

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_{r}}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \le$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_{r}}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \ge 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_{r}}(y)$)

Emulate $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_{r}}(y)$. Before any query of $\mathsf{Sam}^{\pi}_{r}(\mathsf{C})$:
Evaluate $\mathsf{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathsf{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$)

Emulate $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$. Before any query of $\mathsf{Sam}^{\pi}_r(\mathsf{C})$:
Evaluate $\mathsf{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathsf{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathsf{D}}$ is at most twice that of D.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:

$$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$$

$$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$)

Emulate $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$. Before any query of $\mathsf{Sam}^{\pi}_r(\mathsf{C})$:

Evaluate $\mathsf{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathsf{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathsf{D}}$ is at most twice that of D.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathrm{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathrm{hit}^{\pi}_{\mathrm{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathrm{D}}^{\pi,\mathrm{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathrm{hit}^{\pi}_{\widetilde{\mathrm{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathrm{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathrm{D}}^{\pi,\mathrm{Sam}^{\pi}_r}(y)$)

Emulate $\mathrm{D}^{\pi,\mathrm{Sam}^{\pi}_r}(y)$. Before any query of $\mathrm{Sam}^{\pi}_r(\mathrm{C})$:
Evaluate $\mathrm{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathrm{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathrm{D}}$ is at most twice that of D.
- Fix π and y, and let $\delta_i = \Pr_r\left[\mathrm{D}(y)\text{ makes first hit on }i\text{'th Sam query}\right]$.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathsf{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$)

Emulate $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$. Before any query of $\mathsf{Sam}^{\pi}_r(\mathsf{C})$:
Evaluate $\mathsf{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathsf{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathsf{D}}$ is at most twice that of D.
- Fix π and y, and let $\delta_i = \Pr_r\left[\mathsf{D}(y) \text{ makes first hit on } i\text{'th Sam query}\right]$.

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathsf{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathrm{hit}^{\pi}_{\mathsf{D};r}(\pi(x))\right] > \varepsilon\right] \le$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathrm{hit}^{\pi}_{\widetilde{\mathsf{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \ge 0$.**

Proof: We describe a random circuit $\widetilde{\mathsf{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathsf{D}}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$)

Emulate $\mathsf{D}^{\pi,\mathsf{Sam}^{\pi}_r}(y)$. Before any query of $\mathsf{Sam}^{\pi}_r(\mathsf{C})$:
Evaluate $\mathsf{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathsf{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathsf{D}}$ is at most twice that of D.
- Fix π and y, and let $\delta_i = \Pr_r[\mathsf{D}(y)$ makes first hit on i'th Sam query$]$.
  $\implies$ $\Pr_r[\mathsf{D}(y)$ makes first hit on the "x-part" of i'th Sam query$] \ge \delta_i/2$

Proving Claim 20

For any t-augQuery circuit D, $\exists$ $2t$-augQuery circuit $\widetilde{\mathrm{D}}$ such that:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathrm{hit}^{\pi}_{\mathrm{D};r}(\pi(x))\right] > \varepsilon\right] \leq$
$\Pr_{\pi;r}\left[\Pr_x\left[\widetilde{\mathrm{D}}^{\pi,\mathrm{Sam}^{\pi}_r}(\pi(x)) = x \wedge \neg\mathrm{hit}^{\pi}_{\widetilde{\mathrm{D}};r}(p(x))\right] > \varepsilon/2\right]$ **for any $\varepsilon \geq 0$.**

Proof: We describe a random circuit $\widetilde{\mathrm{D}}$, and its deterministic variant follows by fixing the best coins.

Algorithm 22 ($\widetilde{\mathrm{D}}^{\pi,\mathrm{Sam}^{\pi}_r}(y)$)

Emulate $\mathrm{D}^{\pi,\mathrm{Sam}^{\pi}_r}(y)$. Before any query of $\mathrm{Sam}^{\pi}_r(\mathrm{C})$:
Evaluate $\mathrm{C}^{\pi}(z)$ for $z \leftarrow \{0,1\}^n$. If $\mathrm{C}^{\pi}(z)$ makes a query $\pi(x) = y$, return x and halt.

- The augmented query complexity of $\widetilde{\mathrm{D}}$ is at most twice that of D.
- Fix π and y, and let $\delta_i = \Pr_r[\mathrm{D}(y)$ makes first hit on i'th Sam query$]$.
  $\Longrightarrow$ $\Pr_r[\mathrm{D}(y)$ makes first hit on the "x-part" of i'th Sam query$] \geq \delta_i/2$
  $\Longrightarrow$
  $\Pr_r\left[\widetilde{\mathrm{D}}(y) \text{ finds } \pi^{-1}(y) \text{ just before } i\text{'th Sam query, w/o hitting}\right] \geq \delta_i/2.\square$

Proving Claim 21

For any $2^{n/5}/2$-augQuery circuit D:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = x\right] > 2^{-n/5} \land \neg\mathsf{hit}_{\mathsf{D};r}^\pi\right] \leq 2^{-2^{3n/5}}$.

Proving Claim 21

For any $2^{n/5}/2$-augQuery circuit D:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{D}^{\pi,\mathsf{Sam}_r^{\pi}}(\pi(x)) = x\right] > 2^{-n/5} \wedge \neg\mathsf{hit}^{\pi}_{\mathsf{D};r}\right] \leq 2^{-2^{3n/5}}$.

The proof is similar to the non-Sam case.

Proving Claim 21

For any $2^{n/5}/2$-augQuery circuit D:
$\Pr_{\pi;r}\left[\Pr_x\left[\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = x\right] > 2^{-n/5} \wedge \neg\mathsf{hit}^\pi_{\mathsf{D};r}\right] \le 2^{-2^{3n/5}}$.

The proof is similar to the non-Sam case.

Lemma 23 (compression lemma, Sam variant)

For q-augQuery circuit D, $r \in \{0,1\}^*$ and $\varepsilon > 0$, exist algorithms Enc and Dec such that: Let $\pi \in \Pi_n$ be with
$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = x \wedge \neg\mathsf{hit}^\pi_{\mathsf{D};r}(p(x))\right] > \varepsilon$, *then*

- $\mathsf{Dec}(\mathsf{Enc}(\pi)) = \pi$
- $|\mathsf{Enc}(\pi)| \le \log((N-a)!) + 2\cdot\log\binom{N}{a}$, *for* $a \ge \frac{\varepsilon N}{q+1}$

Proving Lemma 23

Proving Lemma 23

Definition 24

Assume $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x))(y)$ makes a query $\mathsf{Sam}_r^\pi(C)$ and get answer (x,x'), we call the π-queries done by $C^\pi(x)$ and $C^\pi(x')$, indirect queries of D.

Construction 25 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon \mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = \pi \wedge \neg \mathsf{hit}^\pi_{\mathsf{D};r}(y)\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lex. element in $\mathcal{I}$.
    1. Add y to $\mathcal{Y}$.
    2. Remove y and all direct & indirect π-queries $\mathsf{D}(y)$ makes from $\mathcal{I}$.

Proving Lemma 23

Definition 24

Assume $D^{\pi,\mathrm{Sam}_r^\pi}(\pi(x))(y)$ makes a query $\mathrm{Sam}_r^\pi(C)$ and get answer (x, x'), we call the π-queries done by $C^\pi(x)$ and $C^\pi(x')$, indirect queries of D.

Construction 25 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon D^{\pi,\mathrm{Sam}_r^\pi}(\pi(x)) = \pi \wedge \neg \mathrm{hit}^\pi_{D;r}(y)\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lex. element in $\mathcal{I}$.
    1. Add y to $\mathcal{Y}$.
    2. Remove y and all direct & indirect π-queries $D(y)$ makes from $\mathcal{I}$.

Algorithm 26 ($\mathrm{Enc}(\pi)$)

Output (description of) $\mathcal{Y}$ and $\mathcal{V} = \{(x, \pi(x)) \colon \pi(x) \notin \mathcal{Y}\}$.

Proving Lemma 23

Definition 24

Assume $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x))(y)$ makes a query $\mathsf{Sam}_r^\pi(C)$ and get answer (x,x'), we call the π-queries done by $C^\pi(x)$ and $C^\pi(x')$, indirect queries of D.

Construction 25 (Useful set $\mathcal{Y} \subseteq \{0,1\}^n$)

1. Set $\mathcal{Y} = \emptyset$ and $\mathcal{I} = \{y \in \{0,1\}^n \colon \mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(\pi(x)) = \pi \wedge \neg \mathsf{hit}^\pi_{\mathsf{D};r}(y)\}$.
2. While $\mathcal{I} \neq \emptyset$, let y be the smallest lex. element in $\mathcal{I}$.
   1. Add y to $\mathcal{Y}$.
   2. Remove y and all direct & indirect π-queries $\mathsf{D}(y)$ makes from $\mathcal{I}$.

Algorithm 26 ($\mathsf{Enc}(\pi)$)

Output (description of) $\mathcal{Y}$ and $\mathcal{V} = \{(x,\pi(x)) \colon \pi(x) \notin \mathcal{Y}\}$.

Under proper encoding, $|\mathsf{Enc}(\pi)| \leq \log((N-a)!) + 2 \cdot \log \binom{N}{a}$ for $a = |Y| \geq \frac{\varepsilon N}{q+1}$.

Proving Lemma 23 cont.

Proving Lemma 23 cont.

Algorithm 27 ($\mathsf{Dec}(\mathcal{Y}, \mathcal{V})$)

For all $y \in \mathcal{Y}$ in lex. order:

1. Emulate $\mathsf{D}^{\pi, \mathsf{Sam}_r^{\pi}}(y)$.
   1. Answer π-query using $\mathcal{V}$.
   2. On Sam-query $\mathsf{Sam}_r^{\pi}(\mathsf{C})$: choose x according to r, and let x' be the first element in $\{0,1\}^n$ for which the π-queries of $\mathsf{C}^{\pi}(x')$ are defined, and $\mathsf{C}^{\pi}(x') = \mathsf{C}^{\pi}(x)$.
2. If D makes a π-query x that is undefined in $\mathcal{V}$, add (x, y) to $\mathcal{V}$.

   Otherwise, add $(\mathsf{D}^{\pi, \mathsf{Sam}_r^{\pi}}(y), y)$ to $\mathcal{V}$.

Use $\mathcal{V}$ to reconstruct π

Proving Lemma 23 cont.

Algorithm 27 ($\mathsf{Dec}(\mathcal{Y}, \mathcal{V})$)

For all $y \in \mathcal{Y}$ in lex. order:

1. Emulate $\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(y)$.
   1. Answer π-query using $\mathcal{V}$.
   2. On Sam-query $\mathsf{Sam}_r^\pi(\mathsf{C})$: choose x according to r, and let x' be the first element in $\{0,1\}^n$ for which the π-queries of $\mathsf{C}^\pi(x')$ are defined, and $\mathsf{C}^\pi(x') = \mathsf{C}^\pi(x)$.
2. If D makes a π-query x that is undefined in $\mathcal{V}$, add (x,y) to $\mathcal{V}$.

   Otherwise, add $(\mathsf{D}^{\pi,\mathsf{Sam}_r^\pi}(y), y)$ to $\mathcal{V}$.

Use $\mathcal{V}$ to reconstruct π

Correctness holds since $\mathsf{hit}_{\mathsf{D};r}^\pi(y) = 0$ for all $y \in \mathcal{Y}$, and thus answer to all Sam-queries are defined.

Remarks

- Results extends to OWFs and to TDPs.

Remarks

- Results extends to OWFs and to TDPs.
- Making Sam use independent randomness per input query C?